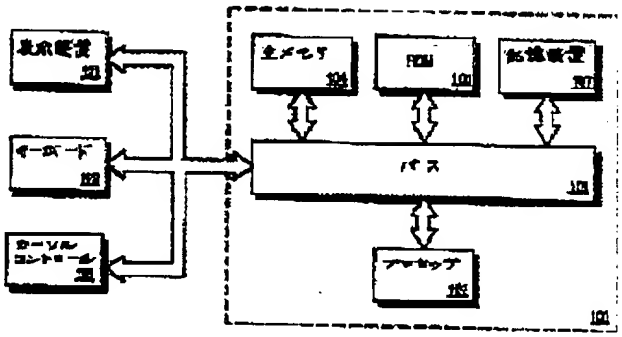
COMPONENT RECORDING METHOD FOR COMPUTER PROGRAM AND DEVICE THEREFOR

Patent number: JP9223024
Publication date: 1997-08-26
Inventor: BAATO SUMAARUDAASU; KEBIN JIEI KURAAKU
Applicant: SUN MICROSYSTEMS INC
Classification:
- international: G06F9/45; G06F9/06
- european: G06F9/445; G06F9/45E3T; G06F9/45M1L; G06F11/34T
Application number: JP19960207692 19960719
Priority number(s): US19950504091 19950719

Also published as:
EP0755003 (A

Abstract of **JP9223024**

PROBLEM TO BE SOLVED: To carry out the same operations with less capacity of a high speed memory by detecting one of plural code sections that is used in an execution period and sequencing these code sections based on the used one. SOLUTION: In the execution of a computer program, a computer system 100 detects specific one of code sections of an instruction data image that is accessed by a processor 102 and also detects the time when the code section is accessed. These access records are stored in an optional storage such as a disk 107 or a memory 104. When the system 100 completes its all writing operations to an access matrix based on the access data stored in an execution period, a binary array corresponding to a specific code section is stored in each row. Then, the order of the code sections is determined based these binary arrays.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－223024

(43)公開日　平成９年(1997)８月26日

(51)Int.Cl.⁶		識別記号	庁内整理番号	ＦＩ			技術表示箇所
Ｇ０６Ｆ	9/45			Ｇ０６Ｆ	9/44	３２２Ｆ	
	9/06	４１０			9/06	４１０Ｅ	

審査請求　未請求　請求項の数７　ＦＤ　（全 13 頁）

(21)出願番号　特願平8－207692

(22)出願日　平成８年(1996)７月19日

(31)優先権主張番号　０８／５０４０９１

(32)優先日　1995年７月19日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　591064003
サン・マイクロシステムズ・インコーポレーテッド
SUN MICROSYSTEMS, INCORPORATED
アメリカ合衆国　94043　カリフォルニア州・マウンテンビュー・ガルシア　アヴェニュウ・2550

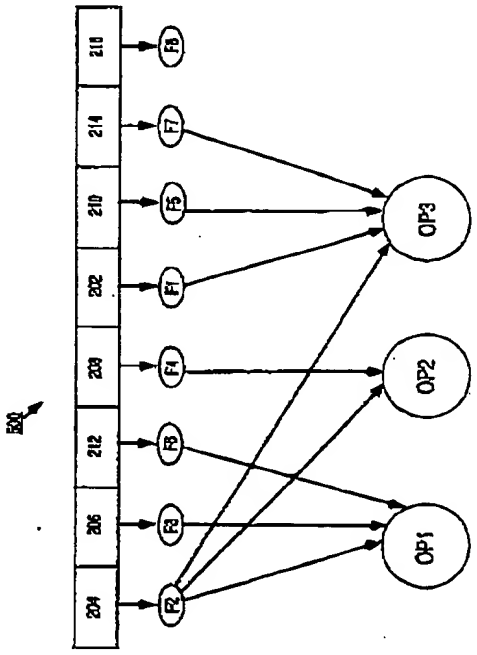
(72)発明者　パート・スマールダース
アメリカ合衆国　95125　カリフォルニア州・サン　ホゼ・ジョナサン　アヴェニュ・1737

(74)代理人　弁理士　山川　政樹

最終頁に続く

(54)【発明の名称】　コンピュータプログラムのコンポーネントを記録するための方法及び装置

(57)【要約】
【課題】　コンピュータプログラムのコードセクションを順序付けるための方法及び装置を提供する。
【解決手段】　実行期間中にコンピュータプログラムを実行し、その実行期間中にコンピュータプログラムの複数の各コードセクションのうちどのコードセクションが使用されたかを検知し、実行期間中に各コードセクションのうちどのコードセクションが使用されたかに基づいてコードセクションを順序付ける。
【効果】　本発明によれば、同じ動作をより少ない高速メモリ容量を用いて実行することが可能になる。



【特許請求の範囲】
【請求項１】　複数のコードセクションを有するコンピュータプログラムのそれらのコードセクションを順序付けるためのコンピュータシステムにおいて実行可能な方法において、
実行期間中にそのコンピュータプログラムを実行するステップと、
その実行期間中に複数の各コードセクションのうちどのコードセクションが使用されたかを検知するステップと、
その実行期間中に各コードセクションのうちどのコードセクションが使用されたかに基づいてコードセクションを順序付けるステップと、を具備した方法。
【請求項２】　前記実行期間を複数の時間間隔に分割するステップと、
前記各コードセクションが使用された複数の時間間隔に基づいてコードセクションを順序付けるステップと、をさらに具備した請求項1記載の方法。
【請求項３】　コンピュータプログラムの複数のコードセクションを記憶するメモリと、
そのメモリに接続されたプロセッサであって、実行期間中に複数のコードセクションにアクセスしてその実行期間中にコンピュータプログラムを実行し、その実行期間中に複数のコードセクションの中の特定のコードセクションがいるプロセッサによってアクセスされたかを指示するアクセス記録を前記メモリに記憶するプロセッサと、
そのアクセス記録に基づいて複数のコードセクションを順序付けるよう構成されたメカニズムと、を具備したコンピュータシステム。
【請求項４】　複数のコードセクションを有するターゲットコンピュータプログラムのそれらのコードセクションを順序付けるためのコンピュータ可読コードが実装されたコンピュータ使用可能媒体と、
ターゲットコンピュータプログラムが実行される実行期間中に複数の各コードセクションが使用された時これをコンピュータに検知させるよう構成されたコンピュータ可読プログラムコードデバイスと、
その実行期間中に各コードセクションが何時使用されたかに基づいてコンピュータにコードセクションを順番付けさせるよう構成されたコンピュータ可読プログラムコードデバイスと、を具備したコンピュータプログラム製品。
【請求項５】　コンピュータに前記実行期間を複数の時間間隔に分割させるよう構成されたコンピュータ可読プログラムコードデバイスと、
その複数の時間間隔のうち各コードセクションが使用された時間間隔に基づいてコンピュータにコードセクションを順番付けさせるよう構成されたコンピュータ可読プログラムコードデバイスと、をさらに具備した請求項4記載のコンピュータプログラム製品。
【請求項６】　コンピュータプログラムによりアクセスされる複数のデータセクションを有するデータを順序付けるコンピュータシステム上で実行可能な方法において、
実行期間中にコンピュータプログラムを実行するステップと、
その実行期間中に複数の各データセクションがコンピュータプログラムによりアクセスされた時これを検知するステップと、
前記実行期間中に各データセクションが何時アクセスされたかに基づいてデータセクションを順序付けるステップと、を具備した方法。
【請求項７】　前記実行期間を複数の時間間隔に分割するステップと、
その複数の時間間隔のうち各データセクションがコンピュータプログラムによってアクセスされた時間間隔に基づいてデータセクションを順序付けるステップと、をさらに具備した請求項6記載の方法。
【発明の詳細な説明】
【０００１】
【発明が属する技術分野】本発明は、ソフトウェアの開発技術に関し、より詳しくは、コンピュータプログラムのコードセクションを仮想メモリ環境における効率的な実行が達成されるよう順序付けるための方法及び装置に関する。
【０００２】
【従来の技術】コンピュータプログラムには、アプリケーションプログラム、デバイスドライバ、ライブラリ等、数多くの種類がある。説明の便宜上、以下本願においては、このような全ての種類のコンピュータプログラムをまとめて、「コンピュータプログラム」と総称する。プログラムの種類に関わらず、あらゆるコンピュータプログラムは、プロセッサによって実行されたとき、プロセッサに何らかの関数を遂行させる命令を含む。コンピュータプログラムが実行されていないときは、命令を表すディジタルデータ（「命令データイメージ」）は通常磁気ディスクまたは光ディスクのような比較的低速の不揮発性記憶装置に記憶されている。記憶装置が低速であることは、コンピュータプログラムの実行時に命令が記憶装置から直接読み出される場合、相当大きなネックになる。
【０００３】このようなネックを回避するために、通常、命令データイメージはプログラムの実行に先立ってランダムアクセスメモリのような比較的高速の揮発性記憶装置にコピーされる。プロセッサは、そのような比較的高速の記憶装置に入れられた命令データイメージのコピーから直接命令を読み込み、これによって低速の記憶装置に常時アクセスする必要が回避される。
【０００４】コンピュータが遂行する関数が複雑になる

にしたがって、命令データイメージのサイズも大きくなる。その結果、プログラム実行時に命令データイメージのコピー全体を記憶するために必要な高速メモリの量も増大する。高速メモリは、比較的高価であるため、プログラムを効率的に実行するのに必要な高速メモリの量を少なくするための技術が開発されてきた。

【０００５】一つの技術によれば、高速記憶装置の利用可能な量よりも大きい「仮想メモリ」を得るために、遅い方の記憶装置の一部を速い方の記憶装置と結合することが行われている。プログラム実行時には、そのプログラムの命令データイメージは仮想メモリに記憶される。通常、このことは、、任意の所与の時点において、命令データイメージのある部分は低速記憶装置に記憶され、命令データイメージの他の部分は高速記憶装置に記憶されるということを意味する。本願においては、高速記憶装置に記憶される命令データイメージの部分をコンピュータプログラムの「ロードされた部分」と称する。高速記憶装置の無駄使いを避けるためには、コンピュータプログラムのロードされた部分を現在の高レベル・オペレーションを実行するのに必要な命令データイメージの部分のみに限定する方が好ましい。

【０００６】典型的には、命令は高速記憶装置にロードしてからでないと実行することができない。そのために、命令を実行する前には、その命令を表す命令データイメージの部分はロードされた部分の一部でなければならない。もしその命令を表す命令データイメージがまだ高速記憶装置に記憶されていなければ、低速記憶装置から高速記憶装置へコピーされる。その時、もし高速記憶装置に十分な未使用空間がなければ、高速記憶装置の一部を必要な命令データイメージを受け入れられるように解放しなければならない。高速記憶装置の一部を解放するプロセスは、高速記憶装置から低速記憶装置への変更されたデータのコピー動作を伴う場合がある。低速記憶装置と高速記憶装置との間でデータが転送される都度、コンピュータプログラムの実行が遅延する。

【０００７】仮想メモリの低速部と高速部との間で行われるデータ転送の回数をできるだけ少なくするための技術が開発されている。具体的に言うと、プロセッサに低レベルオペレーションを行わせる命令は、通常、命令データイメージ中で互いに近くに置かれるということがわかった。従って、一つの命令が実行される時は、近い将来その命令の近くにある他の命令が実行されることが多い。その結果、ほとんどの仮想メモリシステムは、仮想メモリの低速部と高速部との間で、命令単位ではなく、ブロック単位で命令を移動させるようになっている。

【０００８】高レベルオペレーションに対応する命令を強制的に同じコードブロックにまとめることによって、仮想メモリの低速部と高速部との間のデータ転送回数も、ロードされた部分のサイズも小さくすることができる。例えば、関数Ｘ、Ｙ及びＺの実行が必要な独立の高レベルオペレーションＯを考えて見よう。もし、関数Ｘ、Ｙ及びＺを実行するための命令が同じコードブロックにあれば、最小のロードされた部分は単一ブロックからなると思われる。さらに、オペレーションＯの実行前に行われるコードブロック転送は最大１回であろう。しかしながら、関数Ｘ、Ｙ及びＺを実行するための命令シーケンスが各々別個のコードブロックに記憶されている場合は、オペレーションＯの実行時の最小のロードされた部分には３ブロックが含まれることになろう。さらに、ロードされた部分をロードするのに最大３回のコードブロック転送が必要な場合がある。また、関数Ｘのための命令が多くのコードブロックに分散し、関数Ｙのための命令が多くのコードブロックに分散し、関数Ｚのための命令が多くのコードブロックに分散していれば、さらに多くのコードブロックが必要になる場合がある。

【０００９】

【発明が解決しようとする課題】上記の事実に基づいて考えると、最小のロードされた部分のサイズを小さくするための方法及び装置を提供することが望ましい。さらに、仮想メモリ環境におけるコンピュータプログラムの実行時に行われるコードブロック転送の回数を少なくするための方法を提供することが望ましい。さらに、コンピュータプログラムの命令データイメージのセクションを独立の高レベルオペレーションと比較的高い相関を有するグループにまとめるための方法及び装置を提供することが望ましい。

【００１０】

【課題を解決するための手段】本発明は、コンピュータプログラムのコードセクションを順序付けるための方法及び装置である。コンピュータプログラムは試行的実行期間中に実行される。実行期間中にアクセス記録が作成される。そのアクセス記録は、各コードセクションが何時アクセスされたかを指示する。実行期間終了後、コードセクションは、それらが実行期間中いるアクセスされたかに基づいて記録される。

【００１１】本発明の一態様によれば、実行期間は複数の時間間隔に分割される。各時間間隔がコンピュータプログラムの独立の高レベルオペレーションが実行された期間に対応する。そして、各行がコードセクションに対応し、各列が時間間隔に対応するアクセスマトリクスが生成される。対応する時間間隔中に対応するコードセクションがアクセスされたかどうかに基づいて、データマトリクスの各セルに値が記憶される。次に、コードセクションが、対応する行の値に基づいて記録される。

【００１２】本発明のもう一つの態様によれば、アクセスマトリクスの各セルは１つのビットを記憶する。その結果、各コードセクションは対応するビットアレイを有する。コードセクションは、それらの対応ビットアレイによって表される数の数値の順序に基づいて記録することができる。あるいは、コードセクションは、連続する

コードセクションのビットアレイ間の偏差の度合が最小になるようにして記録することも可能である。

【００１３】以下、本発明を添付図面に示す実施形態により説明するが、これらの実施形態は本発明を限定するためのものではない。図中、同じ符号は類似の要素を指す。

【００１４】

【発明の実施の形態】以下、コンピュータプログラムのコードセクションを順序付けるための方法及び装置の実施形態について説明する。以下の説明においては、本発明の完全な理解を図るために、説明の便宜上特定の詳細事項が多数記載されている。しかしながら、本発明がこれら特定の詳細事項の記載なしで実施可能であることは明白であろう。その他の場合においては、周知の構成及び装置は、本発明を不必要にあいまいにすることがないように、ブロック図形式で示されている。

【００１５】図１においては、本発明の実施形態を実装することができるコンピュータシステムが１００で示されている。コンピュータシステム１００は、情報を伝達するためのバスまたはその他の伝送手段１０１、及び情報を処理するためのバス１０１に接続された処理手段１０２を有する。コンピュータシステム１００は、さらに、バス１０１に接続されていて、プロセッサ１０２により実行される情報及び命令を記憶するためのランダムアクセスメモリ（ＲＡＭ）またはその他のダイナミック記憶装置１０４（主メモリと称する）を有する。主メモリ１０４は、プロセッサ１０２による命令実行時に一時的数値変数及び命令を記憶するために使用することもできる。また、コンピュータシステム１００は、バス１０１に接続されていて、プロセッサ１０２用の静的情報及び命令を記憶するためのリードオンリーメモリ（ＲＯＭ）及び／またはその他の静的記憶装置１０６を有する。バス１０１には、情報及び命令を記憶するためのデータ記憶装置１０７が接続されている。

【００１６】磁気ディスクまたは光ディスクのようなデータ記憶装置１０７及びこれに対応するディスクドライブはコンピュータシステム１００に接続することができる。また、コンピュータシステムは、バス１０１を介して、コンピュータユーザに情報を表示するためのブラウン管（ＣＲＴ）のようなディスプレイ装置１２１に接続することもできる。通常は、プロセッサ１０２に情報を伝達し、コマンド選択を行うために、アルファニューメリックキー及びその他のキーを含むアルファニューメリック入力装置１２２がバス１０１に接続される。それ以外の種類のユーザ入力装置として、プロセッサ１０２に対して方向指示情報を入力し、コマンド選択を行うとともに、ディスプレイ１２１上のカーソルの動きを制御するためのマウス、トラックボールあるいはカーソル方向キーのようなカーソルコントロール１２３が接続されている。この入力装置は、通常、平面内のいちを指定することができるよう、２軸、すなわち第１の軸（例えばｘ軸）と第２の軸（例えばｙ軸）における２自由度を有する。

【００１７】あるいは、スタイラスまたはペンのような他の入力装置を用いてディスプレイと対話することもできる。コンピュータスクリーン上に表示されたオブジェクトは、それらの表示されたオブジェクトにタッチするためのスタイラスまたはペンを用いて選択することができる。コンピュータは、その選択されたオブジェクトをタッチ感知スクリーンを用いて検出する。同様に、ライトペンや感光スクリーンを用いて表示されたオブジェクトを選択することもできる。この種の入力装置では、マウスやトラックボールを用いたシステムにおけるように「ポイント・アンド・クリック」操作と異なり、選択位置と選択されたオブジェクトを単一操作として検出することができる。スタイラス及びペン型入力装置やタッチスクリーン及び感光スクリーンは、当技術分野においては周知である。このようなシステムにおいては、インタフェースがすべて筆記具（ペンのような）としてのスタイラスを介して与えられ、書かれたテキストが光学文字認識装置（ＯＣＲ）を用いて変換され、１２２で示すようなキーボードがない場合もある。

【００１８】本発明の一実施形態によれば、コンピュータシステム１００はコンピュータプログラムのコードセクションの順序を決定するよう構成される。この順序は高レベルオペレーションを実行させるコードセクションを一つにまとめさせるものである。その結果、最小にロードされた部分におけるブロック数が少なくなる。さらに、仮想メモリを有するシステムでコンピュータプログラムが実行させるとき起こるコードブロック転送の回数も少なくなる。

【００１９】図２には、一例のコンピュータプログラムＰの命令データイメージ２００が示されている。コンピュータプログラムＰが実行されていないときは、命令データイメージ２００は通常ディスク１０７のような低速記憶装置に記憶される。コンピュータプログラムＰが実行されるときは、命令データイメージ２００は、その一部または全部がメモリ１０４のような高速記憶装置にロードされる。

【００２０】命令データイメージ２００は、コードセクション２０２、２０４、２０８、２１０、２１２、２１４及び２１６を含む。説明の便宜上、コードセクション２０２、２０４、２０８、２１０、２１２及び２１６はそれぞれ関数Ｆ１、Ｆ２、Ｆ３、Ｆ４、Ｆ５、Ｆ６、Ｆ７及びＦ８を実装すると仮定する。また、コンピュータプログラムＰは、独立の３つの高レベルオペレーション、すなわちＯＰ１、ＯＰ２及びＯＰ３を実行するものと仮定する。本願において、「独立のオペレーション」という用語は、いったん呼び出されると、割込みがかけられない限り完遂されるオペレーションを指すものとす

る。このようなオペレーションとしては、例えば、プログラム初期化オペレーション、演算オペレーション及びプログラム終了オペレーション等がある。また、オペレーションＯＰ１は、関数Ｆ２、Ｆ３及びＦ６の実行を必要とし、オペレーションＯＰ２は、関数Ｆ２及びＦ４の実行を必要とし、オペレーションＯＰ３は、関数Ｆ１、Ｆ２、Ｆ５及びＦ７の実行を必要とするものと仮定する。この場合、関数Ｆ８はコンピュータプログラムＰによって実行されるどのオペレーションでも必要とされない。従って、関数Ｆ８を実行するための命令を含むコードセクション２１６は全くアクセスされない。

【００２１】モニタ下におけるプログラムの実行

本発明の一実施形態によれば、コンピュータプログラムＰはコンピュータシステム１００上で実行される。具体的に言うと、ユーザはコンピュータシステム１００にコンピュータプログラムＰの実行を開始させるための入力を入れる。すると、前記の仮定により、ユーザ入力に応答してプログラム初期化オペレーションＯＰ１が実行される。

【００２２】図３には、プロセッサ１０２がコンピュータプログラムＰを実行している期間中に起こる事象のタイムライン３００が示されている。時刻Ｔ０は、ユーザが入力を入れてコンピュータプログラムＰを呼び出す時刻を表している。プロセッサ１０２は、ユーザ入力に応答してオペレーションＯＰ１を実行する。ＯＰ１の実行中、プロセッサ１０２は、それぞれ線３０２、３０４及び３０６によって示すように、コードセクション２０６、２０４及び２１２に入っている命令を読み出し、実行することによって関数Ｆ３、Ｆ２及びＦ６を実行する。

【００２３】図３では、２０６、２０４及び２１２の各コードセクションへのアクセスを１本の線で表してある。しかしながら、１つの関数の実行には、ある特定コードセクションに対するアクセスが何１００回も必要な場合があるということに留意すべきである。一般に、あるコードセクションの最初のアクセスは、そのコードセクションを含むコードブロックを高速記憶装置にロードさせる。同じ独立の高レベルオペレーション中におけるその同じコードへの以後のアクセスは、そのコードセクションを含むコードブロックがその高レベルオペレーションが行われている間高速メモリ外に転送されることがないような高速でもって行われる。そのために、独立の高レベルオペレーションの実行中に特定のコードセクションがアクセスされる回数は、ロードされた部分の最小サイズ及びその高レベルオペレーションの実行中に行わなければならないブロック転送の回数にほとんど影響を及ぼさない。

【００２４】コンピュータプログラムＰの実行中、コンピュータシステム１００は、命令データイメージ２００のどのコードセクションがプロセッサ１０２によってアクセスされたか、及びそれらのコードセクションがアクセスされた時刻を検出する。このデータを以下「アクセス」と称する。コンピュータシステム１００は、アクセスレコードをディスク１０７あるいはメモリ１０４のような任意の記憶装置に記憶することができる。あるいは、コンピュータシステム１００は、ユーザによって何時入力が入れられたかを検出し、記録するようにしてもよい。

【００２５】初期化オペレーションＯＰ１は、プロセッサ１０２が関数Ｆ６の実行を完了すると終了する。オペレーションＯＰ１が終了すると、コンピュータシステム１００は、ユーザが次にどの動作を実行すべきかを指示するまでコンピュータプログラムＰの実行を停止する。コンピュータプログラムＰの実行が停止されている間、プロセッサ１０２は他のコンピュータプログラムに関連する他のオペレーションを実行したり、入力装置をポーリングしてユーザ入力を検出したり、あるいはこれら両方を行うことが可能である。

【００２６】説明の便宜上、時刻Ｔ１にユーザはオペレーションＯＰ２を呼び出すための入力を入れるものと仮定する。コンピュータシステム１００はこのユーザ入力に関連した時刻を検出し、記録する。プロセッサ１０２は、検出されたユーザ入力に応答してオペレーションＯＰ２を実行する。オペレーションＯＰ２の実行中、プロセッサ１０２はコードセクション２０８に入っている命令を読み出し、実行して、関数４を実行し、またコードセクション２０４に入っている関数Ｆ２を実行するための命令を読み出して実行する。コンピュータシステム１００は、これらの各コードセクションが何時アクセスされたかを検出し、アクセスレコードに記録する。オペレーションＯＰ２は、プロセッサ１０２が関数Ｆ２の実行を完了すると終了する。オペレーションＯＰ２が終了すると、コンピュータシステムは次にどのオペレーションを実行すべきかを決めるユーザ入力が供給されるまで待機する。

【００２７】時刻Ｔ２では、ユーザはコンピュータプログラムＰを終了するための入力を入れる。プロセッサ１０２は、このユーザ入力に応答して、コードセクション２０４から命令を読み出し、実行して関数Ｆ２を実行し、コードセクション２１０から命令を読み出し、実行して関数Ｆ５を実行し、コードセクション２０２からコードセクションを読み出し、実行して関数Ｆ１を実行し、コードセクション２１４から命令を読み出し、実行して関数Ｆ７を実行する。コンピュータシステム１００は、これらの各コードセクションのアクセスタイミングをアクセスレコードに記録する。時刻Ｔ３では、命令データイメージ２００に対応するコンピュータプログラムＰの実行が終了する。

【００２８】アクセス検出

上記のモニタ下におけるプログラム実行プロセス時に

は、コンピュータシステム１００は、各特定のコードセクションがアクセスされたとき、これを検出して記録する。このアクセス検出を行うためには多くの技術的方法を用いることができる。本発明の一実施形態によれば、各コードセクションには命令シーケンスが入れられる。命令シーケンスは、プロセッサ１０２にその命令シーケンスがあるコードセクションを識別指示するデータ及び各時点における現在時刻を記録させる。例えば、コードセクション２０２には、実行されると、プロセッサ１０２に関数Ｆ１を識別指示するデータ及び現在時刻を記録させる命令シーケンスを埋め込むことができる。同様に、コードセクション２０４には、実行されると、プロセッサ１０２に関数Ｆ２を識別指示するデータ及び現在時刻を記録させる命令シーケンスを埋め込むことができる。

【００２９】本発明のもう一つの実施形態においては、コンピュータシステム１００は、コンピュータプログラムＰの実行と同時に検出プログラムを実行するよう構成される。検出プログラムは、コンピュータシステム１００にプロセッサがコンピュータプログラムＰのいろいろなコードセクションにアクセスしているとき、それを検出、記録させる。本発明のさらに他の実施形態においては、コンピュータプログラムＰの実行時にプロセッサ１０２によってなされたコールを検出、記録するためのモニタ用ハードウェアがコンピュータシステム１００に付加される。

【００３０】アクセスマトリクスの構築

コンピュータプログラムＰの実行が終了すると、コンピュータシステム１００は、コンピュータプログラムＰの実行中に記憶された情報に基づいてアクセスマトリクスをメモリ１０４内に構築する。

【００３１】最初に、コンピュータシステム１００はＴ０とＴ３の間の期間（「実行期間」）をいくつかの時間間隔に分割する。実行期間を分割するこれらの時間間隔の大きさ、数、及び境界を決定するには種々の方法を用いることができる。例えば、実行期間は所定の持続時間を有する時間間隔に分割することができる。又は、実行期間は、同じ持続時間の所定数の時間間隔に分割することもできる。また、実行期間は、実行期間中の関数呼出しの密度と分布及び／またはユーザ入力のタイミングのような要素に基づく可変持続時間の時間間隔に分割することもできる。本発明は、実行期間を幾つかの時間間隔に分割する各特定の技術的方法に限定されるものではない。しかしながら、これにより選択される時間間隔は、各時間間隔内で低い平均ページング速度が確保されるよう十分短くなることが望ましい。

【００３２】この例においては、コンピュータシステム１００は、実行期間を実行期間中のユーザ入力のタイミングに基づいて時間間隔に分割する。具体的に言うと、実行期間は３つの時間間隔ＩＮＴ１、ＩＮＴ２及びＩＮＴ３に分割される。時間間隔ＩＮＴ１は、プログラムＰを呼び出した入力とオペレーションＯＰ２を呼び出した入力との間の時間に広がる。時間間隔ＩＮＴ２は、オペレーションＯＰ２を呼び出した入力とオペレーションＯＰ３を呼び出した入力との間の時間に広がる。時間間隔ＩＮＴ３は、オペレーションＯＰ３を呼び出した入力とコンピュータプログラムＰの終了との間の時間に広がる。

【００３３】これらの種々の時間間隔が決定されたならば、コンピュータシステム１００は、各行が命令データイメージ２００中のコードセクションに対応し、各列が１つの時間間隔に対応するマトリクスを構築する。マトリクスは、例えば、メモリ１０４中に構築することができる。図４には、それぞれ時間間隔ＩＮＴ１、ＩＮＴ２及びＩＮＴ３に対応する３本の列と、それぞれコードセクション２０２、２０４、２０６、２０８、２１０、２１２、２１４及び２１６に対応する８本の行４０８、４１０、４１２、４１４、４１６、４１８、４２０及び４２２を有するアクセスマトリクスが示されている。

【００３４】コンピュータシステム１００は、実行期間中に生成されたアクセスレコードに基づいてマトリクス４００に値を書き込む。一実施形態によれば、アクセスマトリクス４００の各セルには１ビットが書き込まれる。各セルのビットは、各セルが存在する行に対応するコードセクションが、そのセルが存在する列に対応する時間間隔中にアクセスされた場合にのみセットされる。例えば、セル４２４はコードセクション２０２に対応する行４０８、及び時間間隔ＩＮＴ１に対応する列４０２にある。その結果、セル４２４に記憶されたビットは、コード２０２が時間間隔ＩＮＴ１中にアクセスされた場合にのみセットされる。上記の例においては、コードセクション２０２は時間間隔ＩＮＴ１中にはアクセスされたなかったので、セル４２４には”０”が書き込まれる。

【００３５】コンピュータシステムが実行期間中に記憶されたアクセスデータに基づいてアクセスマトリクス４００に全て書き込み終えると、各行には特定のコードセクションに対応する２進アレイが記憶された状態になる。例えば、コードセクション２０２に対応する行４０８には、２進アレイ”００１”が記憶される。次に、コンピュータシステム１００は、各コードセクションに対応する２進アレイに基づいてそれらのコードセクションの順序を決定する。

【００３６】順序決定

各コードセクションに対応する２進アレイに基づいてそれらのコードセクションの順序を決定するには、いろいろな技術的方法を用いることができる。例えば、各コードセクションに対応する２進アレイによって表される２進数の値に基づいて各コードセクションにある数を割り当てることができる。この順序付け方法をここで説明し

ている例に適用すると、コードセクション２０２には、その2進アレイは“００１”であるから、数“1”が割り当てられる。コードセクション２０４には、このコードセクション２０４に対応する2進アレイ“１１１”によって表される数“7”が割り当てられる。また、コードセクション２０６、２０８、２１０、２１３、２１４及び２１６には、それぞれ数４、２、１、４、１及び０が割り当てられる。

【００３７】各コードセクションに数が割り当てられたならば、コンピュータシステム１００は各コードセクションに割り当てられた数の大きさの順に、コードセクションの順序を決定する。例えば、数“7”が割り当てられたコードセクション２０４の次に数“4”が割り当てられたコードセクション２０６及び２１２が来る。コードセクション２０６及び２１２の後には、数“2”が割り当てられたコードセクション２０８が来、その後に、数“1”が割り当てられたコードセクション２０２、２１０及び２１４が来る。数“0”が割り当てられたコードセクション２１６はこの順番の最後になる。

【００３８】ここで説明する例においては、複数のコードセクションが同じ2進アレイを有する。例えば、コードセクション２０２、２１０及び２１４は全て同じ2進アレイ“００１”を有する。そのために、これらのコードセクションを互いにどのように順序付けるかを決定するには2進アレイ以外の何かを利用しなければならない。一実施形態によれば、同じ2進アレイを有するコードセクション間の順序はランダムに行われるだけである。もう一つの実施形態においては、同じ2進アレイを有するコードセクションは、それらのコードセクションが1つまたは2つ以上の時間間隔中にアクセスされた順序に基づいて順序付けられる。例えば、コードセクション２０２、２１０及び２１４は全て時間間隔ＩＮＴ３中にアクセスされている。これらの3つのコードセクションの中では、コードセクション２１０が最初にアクセスされ、コードセクション２０２が次にアクセスされ、コードセクション２１４が3番目にアクセスされる。従ってこのアクセス順をコードセクションの順序付けに反映することができる。

【００３９】図5は、上記のプロセスによって決定された順序に従い命令データイメージ２００のコードセクションを順序付けることにより構築された命令データイメージ５００を示す。ここで、命令データイメージ５００のコードセクションの位置とそれらのコードセクションを使用する動作との間には、命令データイメージのコードセクションの位置とそれらのコードセクションを使用する動作との間よりも強い相関があるということに留意すべきである。このようなより強い相関の結果、命令データイメージによって表されるコンピュータプログラムＰが仮想メモリシステムで実行される場合、相当大きな性能の改善が達成される。

【００４０】例えば、２０２、２０４、２０６、２０８、２１０、２１２、２１４及び２１６の各コードセクションのサイズは１Ｋバイトであると仮定する。また、コンピュータプログラムＰは、仮想メモリの低速部と高速部との間で４Ｋブロック単位で転送を行う仮想メモリシステム上で実行されるものと仮定する。これらの条件下では、８つのコードセクションは2つの４Ｋブロックにまたがって分布する。コードセクション２０２、２０４、２０６、２０８、２１０、２１２、２１４及び２１６が命令データイメージ２００に示されているように順序付けられるとすると、第１ブロック（ブロック１）には、コードセクション２０２、２０４、２０６及び２０８が入り、第２ブロック（ブロック２）には、コードセクション２１０、２１２、２１４及び２１６が入ることになる。コードセクション２０２、２０４、２０６、２０８、２１０、２１２、２１４及び２１６が命令データイメージ５００に示されているように順序付けられるとすると、第１ブロック（ブロックＡ）には、コードセクション２０４、２０６、２１２及び２０８が入り、第２ブロック（ブロックＢ）には、コードセクション２０２、２１０、２１４及び２１６が入ることになる。

【００４１】上記に基づいて、コードセクションが命令データイメージ２００に示されているように順序付けられとすると、オペレーションＯＰ１の実行には2つの４Ｋブロック（ブロック１及びブロック２）のローディングが必要であり、オペレーションＯＰ２の実行には1つの４Ｋブロック（ブロック１）のローディングが必要であり、オペレーションＯＰ３の実行には2つの４Ｋブロック（ブロック１及び２）のローディングが必要である。これに対して、コードセクションが命令データイメージ５００に示されているように順序付けられるとすると、オペレーションＯＰ１を実行するには１つのブロック（ブロックＡ）だけロードすればよく、オペレーションＯＰ２を実行するには１つのブロック（ブロックＡ）をロードするだけでよく、オペレーションＯＰ３を実行するには２つのブロック（ブロックＡ及びブロックＢ）をロードするだけでよい。このようにして、コードセクションが仮想メモリの高速記憶部に入っていない場合、仮想メモリの低速記憶部から高速記憶部にコピーすることによってコードセクションが「ロードされる」。

【００４２】コードセクションが命令データイメージ５００に示されているように順序付けられると、初期化オペレーションＯＰ１実行時にロードするブロックが１つ少なくて済むことによって、相当大きい効率改善が達成される場合がある。例えば、２ブロック初期化オペレーションＯＰ１の場合に必要になる可能性のある余分のオペレーションについて考えてみよう。仮想メモリの高速部がすでにいっぱいに書き込まれているとすると、ブロック２を書き込むためのスペースを得るために１ブロックのデータを仮想メモリの低速記憶部に書き込まなけれ

ばならない場合がある。その場合、ＯＰ１が終了すると、ブロック２がまさしく高速記憶部の貴重なスペースを、割り当て解除されるまで、あるいはコンピュータプログラムＰが終了するまで占有することになる。

【００４３】ここで、簡単な例を取り上げて、本発明をわかりやすく説明した。しかしながら、多くのコンピュータプログラムはここで説明するコンピュータプログラムＰの例よりかなり複雑である。実際、多くのコンピュータプログラムは、数百の独立した高レベルオペレーション及び数千の関数を実行する。このような複雑な条件においては、本発明によりコードセクションを順序付けることによってより大きな効率改善を達成することができる。例えば、複雑な動作を行うコードセクションは、それらのコードセクションを順序付ける前に１０の４Ｋデータブロックにわたって分布する場合がある。そして、本発明の順序付けによって、コードセクションを１つの４Ｋデータブロックにまとめることができる。このように、上記のオペレーションを実行するには、高速記憶部に１０分の１の数のデータブロックをローディングしさえすればよい。

【００４４】ビットパターンの類似性に基づく順序決定上に説明した順序付け方法は、アクセスレコードに基づいてコードセクションを順序付けるのに可能な数多くの方法の中の１つでしかない。本発明のもう一つの実施形態によれば、アクセスマトリクス４００は上に説明したようにして構築されるが、コードセクションは、連続するコードセクションのビットアレイ間でできるだけ少ない数のビットが異なるようにして順序付けられる。

【００４５】このような順序に達するには、最初のコードセクションを選択し、その新しいコードセクション順序の最初に置く。この第１のコードセクションに対応するビットアレイが「現在のビットアレイ」になる。この現在のビットアレイを他のコードセクションのビットアレイと比較して、現在のビットアレイとの差違が最も小さいビットアレイを決定する。この場合、２つのビットアレイで、対応するビット値が何ビット異なるかによってそれら２つのビットアレイ間の差違度が決定される。例えば、コードセクション２０２のビットアレイ“００１”中の３ビットはコードセクション２１０のビットアレイ“００１”の３ビットと同じである。その結果、これら２つのビットアレイ間の差違度はゼロである。これに対して、コードセクション２０２のビットアレイ“００１”は、第２ビットだけがコードセクション２０６のビットアレイ“１００”の対応ビットと一致する。従って、コードセクション２０２と２０６のビットアレイ間の差違度は２である。差違度は、２つのビットアレイの排他的ＯＲでセットされるビット数をカウントすることにより得ることができる。

【００４６】現在のビットアレイとの差違度が最も小さいビットアレイが決まると、それが現在のビットアレイになり、そのビットアレイに対応するコードセクションが新しいコードセクション順序の２番目の位置に置かれる。次に、その新しい現在のビットアレイが新しいコードセクション順序にまだ入れられていない全てのコードセクションのビットアレイと比較される。そして現在のビットアレイとの差違度が最も小さいビットアレイが新しい現在のビットアレイになり、そのビットアレイに対応するコードセクションが新しいコードセクション順序の３番目の位置に置かれる。このプロセスが、全てのコードセクションが新しいコードセクション順序に並べられるまで繰り返される。

【００４７】この順序付け方法をここで説明する例に適用する場合、数値的に最も高位のビットアレイを有するコードセクション２０４が新しいコードセクション順序の最初のコードセクションとして選択されるものと仮定する。これ以外の基準を用いて第１のコードセクションを選択することも可能である。例えば、最初のコードセクションは、無作為に選択することもできれば、プログラム開始時に実行される最初のコードセクションとなるように選択することもでき、あるいは数値的に最も下位のビットアレイとなるように選択することもできる。本発明は、このような特定の選択基準に限定されるものではない。

【００４８】コードセクション２０４を新しいコードセクション順序の最初のコードセクションとして選択すると、コードセクション２０４に対応するビットアレイ“１１１”が現在のビットアレイになる。この現在のビットアレイを他のビットアレイと比較して、他の各ビットアレイとの間の差違度を求める。そして、現在のビットアレイとの差違度が最も小さいビットアレイに対応するコードセクションが新しいコードセクション順序における次のビットアレイとして選択される。“１１１”と他の全てのビットアレイとの間の差違度は、コードセクション２１６に対応する差違度３のビットアレイ“０００”を除いて、２である。

【００４９】新しいコードセクション順序におけるコードセクション２０２のすぐ後には１つのコードセクションしか来ることができない。そのために、同じ差違度を有するビットアレイ間の「タイ」を解消するための何らかの基準を導入しなければならない。これらのタイを解消するための一つの方法は、タイに関わるコードセクションの１つを無作為に選択することであろう。しかしながら、タイに関与するコードセクションの中で、実行期間中に最初にアクセスされたコードセクションを選択することによってより良い結果を得ることができる。ここで説明する例においては、コードセクション２０６が実行期間中にアクセスされた最初のコードセクションである。従って、コードセクション２０６が新しいコードセクション順序の２番目のコードセクションとして選択される。

【００５０】コードセクション２０６がコードセクション順序における次のコードセクションとして選択されると、コードセクション２０６に対応するビットアレイ“１００”が現在のビットアレイになる。この現在のビットアレイが、まだ新しいコードセクション順序に入っていない全てのコードセクションのビットアレイと比較されて、現在のビットアレイとこれら他のビットアレイとの間の差違度が求められる。そして、現在のビットアレイとの差違度が最も小さいビットアレイに対応するコードセクションが新しいコードセクション順序における次のビットアレイとして選択される。

【００５１】ここで説明する例においては、コードセクション２１２に対応するビットアレイ“１００”は現在のビットアレイと同じである。従って、コードセクション２１２が新しいコードセクション順序の次のコードセクションとして選択される。その結果、コードセクション２１２に対応するビットアレイ“１００”が現在のビットアレイになる。

【００５２】この現在のビットアレイが新しいコードセクション順序にまだ入っていない全てのコードセクションのビットアレイと比較されて、現在のビットアレイとこれら他のビットアレイとの間の差違度が求められる。そして、現在のビットアレイとの差違度が最も小さいビットアレイに対応するコードセクションが新しいコードセクション順序の次のビットアレイとして選択される。

【００５３】ここで説明する例においては、コードセクション２１６に対応するビットアレイ“０００”が現在のビットアレイとの差違度が最も小さい。従って、コードセクション２１６が新しいコードセクション順序における次のコードセクションとして選択される。その結果、コードセクション２１６に対応するビットアレイ“０００”が現在のビットアレイになる。

【００５４】この現在のビットアレイ“０００”が新しいコードセクション順序にまだ入っていない全てのコードセクションのビットアレイと比較されて現在のビットアレイとこれら他のビットアレイとの間の差違度が求められる。そして現在のビットアレイとの差違度が最も小さいコードセクションが新しいコードセクション順序における次のビットアレイとして選択される。

【００５５】ここで説明する例においては、コードセクション２０２、２０８、２１０及び２１４に対応するビットアレイは現在のビットアレイとの差違度が全て同じである。これらのコードセクションの中では、コードセクション２０８が実行期間中にアクセスされた最初のコードセクションである。そのために、コードセクション２０８が新しいコードセクション順序における次のコードセクションとして選択される。その結果コードセクション２０８に体操するビットアレイ“０１０”が現在のビットアレイになる。

【００５６】この現在のビットアレイ“０１０”が新しいコードセクション順序にまだ入っていない全てのコードセクションのビットアレイと比較されて、現在のビットアレイとこれら他のビットアレイとの間の差違度が求められる。そして、現在のビットアレイとの差違度が最も小さいビットアレイに対応するコードセクションが新しいコードセクション順序における次のビットアレイとして選択される。

【００５７】ここで説明する例においては、コードセクション２０２、２１０及び２１４に対応するビットアレイは現在のビットアレイとの差違度が全て同じである。これらのコードセクションの中では、コードセクション２１０が実行期間中にアクセスされた最初のコードセクションである。従って、コードセクション２１０が新しいコードセクション順序における次のコードセクションとして選択される。その結果、コードセクション２１０に対応するビットアレイ“００１”が現在のビットアレイになる。

【００５８】この現在のビットアレイ“００１”が新しいコードセクション順序にまだ入っていない全てのコードセクションのビットアレイと比較されて、現在のビットアレイとこれら他のビットアレイとの間の差違度が求められる。そして、現在のビットアレイとの差違度が最も小さいビットアレイに対応するコードセクションが新しいコードセクション順序における次のビットアレイとして選択される。

【００５９】ここで説明する例においては、コードセクション２０２及び２１４に対応するビットアレイは現在のビットアレイと同じである。これらのコードセクションの中では、コードセクション２０２が実行期間中にアクセスされた最初のビットアレイである。従って、コードセクション２０２が新しいコードセクション順序における次のコードセクションとして選択される。その結果、コードセクション２０２に対応するビットアレイ“００１”が現在のビットアレイになる。

【００６０】この現在のビットアレイ“００１”が新しいコードセクションにまだ入っていない全てのコードセクションのビットアレイと比較されて、現在のビットアレイとこれら他のビットアレイとの間の差違度が求められる。そして、現在のビットアレイとの差違度が最も小さいビットアレイに対応するコードセクションが新しいコードセクション順序における次のビットアレイとして選択される。

【００６１】この時点になると、コードセクション２１４が新しいコードセクション順序に入っていない唯一のコードセクションになる。その結果、コードセクション２１４が新しいコードセクション順序における次のコードセクションとして選択される。これで、全てのコードセクションが新しいコードセクション順序に並べられて、順序付けプロセスは終了する。

【００６２】図６は、コードセクションが上記の「最小

差違度」順序付けプロセスの結果により順序付けられたコンピュータプログラムPの命令データイメージ６００を示す。命令データイメージ５００の場合と同様に、図示の命令データイメージ６００のコードセクションの順序は、元の命令データイメージ２００におけるコードセクションの順序に比べて、コンピュータプログラムPによって実行される高レベルオペレーションに対してより強い相関を有する。

【００６３】上には、実行期間中にコードセクションが何時アクセスされるかに基づいてコードセクションを順序付ける２つの方法を説明した。しかしながら、上に説明した方法は、単に例示的なものである。本発明においては、これら以外にもいろいろな方法を用いることができる。例えば、コードセクションをそれらのコードセクションが実行期間中に最初にアクセスされる順序を反映するよう順序付けることもできる。ここで説明する例にこの方法を適用すると、コードセクションは次の順序により順序付けされる。コードセクション２０６、コードセクション２０４、コードセクション２１２、コードセクション２０８、コードセクション２１０、コードセクション２０２、コードセクション２１４、そして最後にコードセクション２１６。この順序はアクセスレコードから容易に決定することができ、アクセスマトリクスの構築を必要としないということに留意すべきである。

【００６４】命令データイメージの順序付け

コンピュータプログラムのコードセクションの新しい順序が決定されたならば、その新しいコードセクション順序を反映するようコードセクションが再配列された新しい命令データイメージの構築が行われる。本発明の一実施形態によれば、新しい命令データイメージはマップファイルに基づいて生成される。具体的に言うと、Ｆ１、Ｆ２、Ｆ３、Ｆ４、Ｆ５、Ｆ６、Ｆ７及びＦ８の各関数に対応するオブジェクトコードファイルのリンク順序を指定するマップファイルが生成される。各関数に対して指定されるリンク順序は、それらの関数に対応するコードセクションについて決定された順序に基づいて指定される。従って、図６に示す命令データイメージを構築するためには、マップファイルは、オブジェクトコードファイルがＦ２、Ｆ３、Ｆ６、Ｆ８、Ｆ４、Ｆ５、Ｆ１、Ｆ７の順序に従ってリンクすべきであるということを指定する。このマップファイルはプロセッサ１０２で実行中のリンカープログラムによって読み取られる。リンカープログラムは、マップファイルに基づいて、プロセッサ１０２に関数Ｆ１、Ｆ２、Ｆ３、Ｆ４、Ｆ５、Ｆ６、Ｆ７及びＦ８に対応する種々のオブジェクトコードモジュールをリンクさせて、図６に示す命令データイメージ６００を構築する。

【００６５】本発明により決定されるコードセクション順序に基づいてコードセクションが順序付けられた命令データイメージを構築するには、他の方法を用いることもできる。例えば、コンパイラマップファイルを読み取るためのコンパイラを構築することもできる。その場合、コンパイラマップファイルは、関数または関数の部分に対してオブジェクトコードが生成される順序を指定する。あるいは、現命令データイメージを単に再配列するだけで新しい命令データイメージを形成するメカニズムを構築することもできる。例えば、プロセッサ１０２は、命令データイメージ２００からコードセクションを抽出して、命令データイメージ６００を生成するための新しい順序に従い記憶するようプログラムすることができる。この再配列プロセスは、異なるコードセクション内における一部のアドレス参照を参照されるコードの新しい位置を反映するよう修正することが必要な場合もある。

【００６６】以上の説明においては、コードセクションとの関連で順序付け方法を特定的に説明した。しかしながら、本発明はコードセクションの順序付けに限定されるものではない。具体的に言うと、上に説明した技術的方法は、プログラムのコードを順序付ける代わりに、コンピュータプログラムの実行中にアクセスされるデータを順序付けるために用いることもできる。例えば、データベースプログラムにおいては、プログラムはデータの一部のセクションに他のセクションとは異なるタイミング及び周波数でアクセスすることが可能である。これらのデータアクセスは、データのセクションがアクセスされるのと同じ順序で記憶されている場合により効率的である。この順序付け方法をデータに適用するには、データベースプログラムは実行期間中に実行される。そして、実行期間中にデータベースプログラムによって行われたデータアクセスが検出され、記録される。このように記録されたアクセス情報を用いて、データが上記の順序付け方法に従い順序付けられる。

【００６７】プログラムされた「オブジェクト」は、関連付けられた関数及びデータの集合である。上記の技術的方法を用いて全コードまたは全データのセクションを順序付けることができるのとまったく同様に、上記の方法は、コードとデータをどちらも含むオブジェクトを順序付けるために用いることもできる。

【００６８】

【発明の効果】本願で説明した順序付け方法を用いることによって、明確かつ顕著な２つの効果を達成することができる。第１に、同じ動作に対応するルーチンのいくつかのコード部分をより緊密な関係にまとめることができる。所与の動作に対するコードがより少ないコードセクションにまたがって分布することになるため、所与の動作の実行時に高速メモリに常駐させなければならない平均コード量が少なくなる。その結果、同じ動作をより少ない高速メモリ容量を用いて実行することができる。これによって、より少ない高速メモリを持つシステムは、順序付けられたコードを、より多量の高速メモリが

順序付けられていないコードを実行するのと同じ効率レベルで実行することができる。さらに、他の同時実行される動作のためにより多くの高速メモリが使用可能になる。

【００６９】もう一つの効果は、動作を遂行するのに必要なコードセクションをローディングするとき起こるページフォールトの平均数が著しく少なくなる。ページフォールト数が少なくなるのは、コードセクションが使用される順にそろえられるためである。従って、いっしょに用いられ、いっしょにロードされるコードセクションは一つまとめて記憶される。

【００７０】以上、本発明の特定の実施形態について詳細に説明したが、本願によって様々な修正態様及び変形態様は自明であろう。そのような修正態様並びに変形態様は、特許請求の範囲に記載するところに従い本発明の範囲内に包含されるものである。

【図面の簡単な説明】

【図１】　本発明を実装することができるコンピュータシステムのブロック図である。

【図２】　一例のコンピュータプログラムの異なるコードセクションを示す説明図である。

【図３】　図２に示すコンピュータプログラムの異なるコードセクションが実行期間中に何時アクセスされるかを示すタイミング図である。

【図４】　図２に示す実行期間中に記録されたアクセスに基づいて構成されたアクセスマトリクスを示す説明図である。

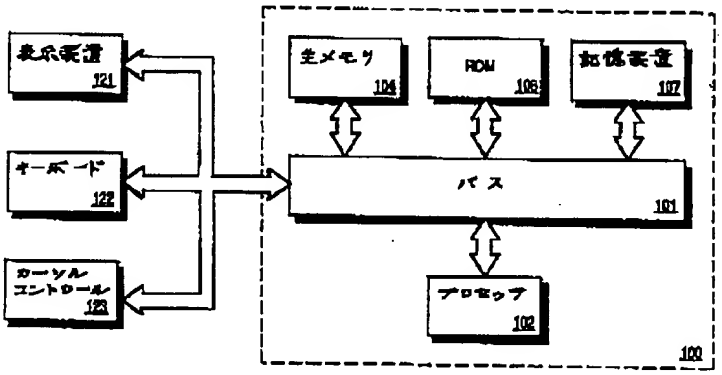
【図５】　図２に示すコンピュータプログラムのコードセクションが、本発明の一実施形態に従って記録された新規な命令データイメージを示す説明図である。

【図６】　図２に示すコンピュータプログラムのコードセクションが、本発明の一実施形態に従って記録された新規な命令データイメージを示す説明図である。

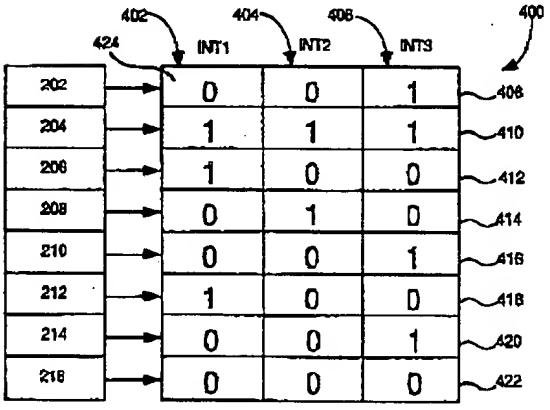
【符号の説明】

１００　コンピュータシステム、１０１　バス、１０２　プロセッサ、１０４主メモリ、１０６　ＲＯＭ、１０７　記憶装置、１２１　表示装置、１２２キーボード、１２３　カーソル制御。

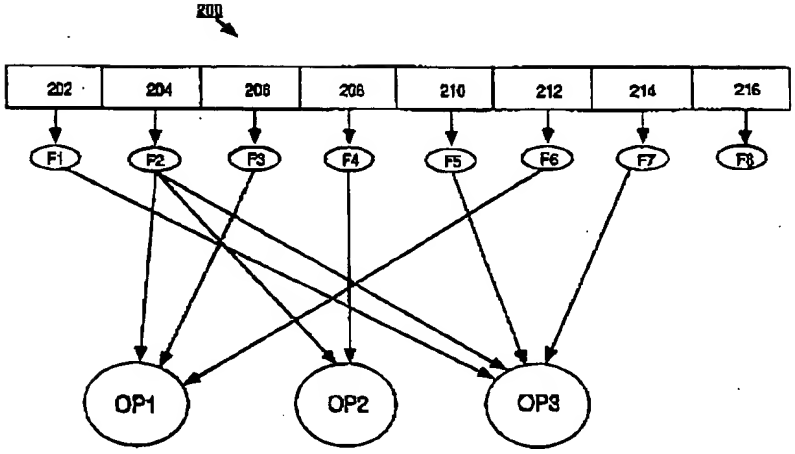
【図１】



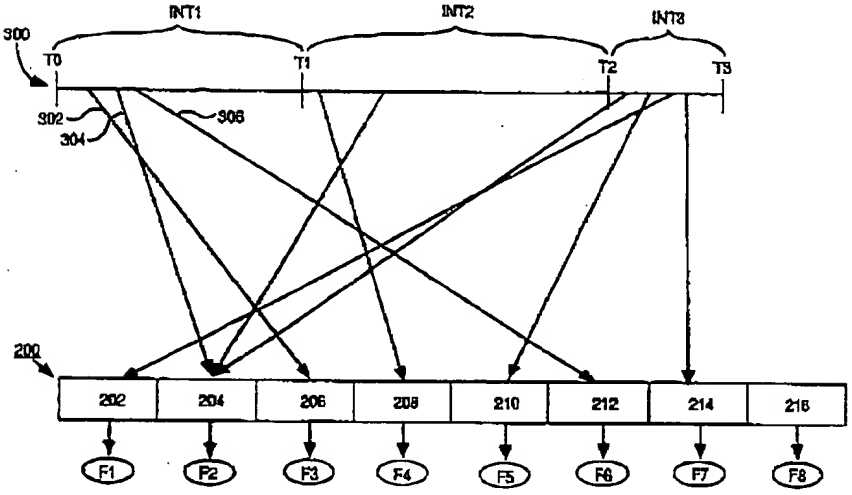
【図４】



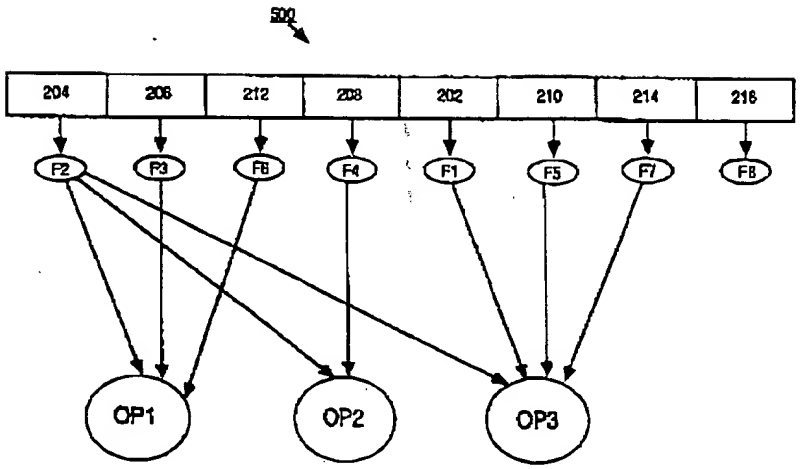
【図2】


200
202
204
206
208
210
212
214
216
F1
F2
F3
F4
F5
F6
F7
F8
OP1
OP2
OP3

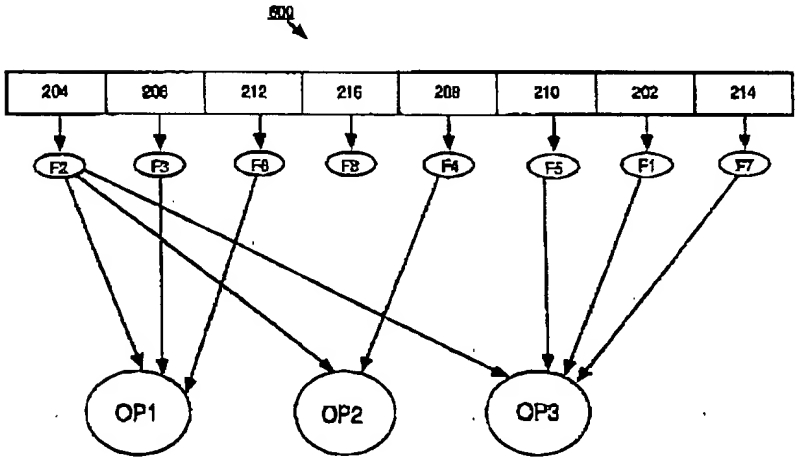
【図3】


INT1
INT2
INT3
300
T0
T1
T2
T3
302
304
306
200
202
204
206
208
210
212
214
216
F1
F2
F3
F4
F5
F6
F7
F8

【図5】



【図6】



フロントページの続き

(72)発明者　ケビン・ジェイ・クラーク
アメリカ合衆国　95125　カリフォルニア州・サン　ホゼ・カーティス　アヴェニュ・1376